# An Ontological Approach for Recovering Legacy Business Content

Aseem Daga, Sergio de Cesare, Mark Lycett and Chris Partridge
DISC, Brunel University
Uxbridge, Middlesex, UK
Firstname.lastname@brunel.ac.uk

***Abstract*— Legacy Information Systems (LIS) pose a challenge for many organizations. On one hand, LIS are viewed as aging systems needing replacement; on the other hand, years of accumulated business knowledge have made these systems mission-critical. Current approaches however are often criticized for being overtly dependent on technology and ignoring the business knowledge which resides within LIS. In this light, this paper proposes a means of capturing the business knowledge in a technology agnostic manner and transforming it in a way that reaps the benefits of clear semantic expression – this transformation is achieved via the careful use of ontology. The approach called Content Sophistication (CS) aims to provide a model of the business that more closely adheres to the semantics and relationships of objects existing in the real world. The approach is illustrated via an example taken from a case study concerning the renovation of a large financial system and the outcome of the approach results in technology agnostic models that show improvements along several dimensions.**

***Keywords- legacy system; legacy transformation; ontology; content transformation.***

## I. INTRODUCTION

The software engineering field has evolved and matured over the last four decades, but the so called 'Legacy Information System' (LIS) problem still exists [1]. Monolithic systems, designed and developed during the 1960s and 1970s, continue to operate and perform within a large number of organizations. These systems run on obsolete hardware and software capabilities, are designed in a stovepipe fashion, have rigid work processes and their maintenance budgets alone consume around 60-80% of the software related budgets of the organization [2]. Constant restructuring and alteration is often carried out to keep them functionally operational, however lack of system knowledge and appropriate documentation has meant that certain systems continue to run on their original specification. While such systems are often viewed as prime candidates for replacement; having been in operation for a number of years, these systems have also become mission-critical to the organization. Not only do they perform important functions for the organization but in some situations they are the only source of business knowledge (for example in the form of business rules, work processes and more importantly business data) [3] [4]. Systems that exhibit such a dual nature are known as LIS, and are prominent within a number of organizations.

The duality associated with LIS has made looking for appropriate solutions a difficult exercise. While a number of approaches have emerged that aim to resolve the LIS problem [5] [6] [2] [7] [8], they remain largely ineffective in providing a stable solution [9] [10]. The major issue with current approaches is that they view the LIS problem primarily from a technical perspective. The limitation with such a perspective is that it not only ignores the organizational and business aspects of LIS, but it also means that the whole effort is spent on creating a LIS for the future[1]. The business knowledge, often vital for the organization, is either neglected or always tied into some technology. As a result, while on one hand, the program understanding task becomes that much more difficult and costly to implement, on the other hand, it also means that little or no effort is spent on preserving or sophisticating (improving) the business knowledge in a manner that allows for its repeated use.

Given the increasing need to identify the enduring and stable aspects of the system [11], this paper stresses the need to capture and document such business knowledge into business models, where the knowledge can be understood and reflected through a business perspective and allowed to evolve in line with business needs [12]. This has two distinct advantages. Firstly, it can deliver a system whose business knowledge is provably more sophisticated than that of the pre-existing system and secondly, since the approach is model-driven, the achieved benefits outlive the individual project and can be re-employed in subsequent renovation projects.

To achieve the desired benefits, this paper presents an *ontology-based* approach for renovating LIS called Content Sophistication (CS). The approach is described in the context of an example of a large financial organization to show (a) that the practical benefits of applying the CS approach are significant and (b) models that emerge from the CS process are semantically rich and are stable across contexts of use. In an effort to clearly distinguish the ontological aspect from its epistemic counterpart, the present research focuses on extracting the knowledge from the data residing within the LIS. The main objective of CS is to extract and document the business knowledge from the business data and interpret and improve its real-world semantics (clarify its knowledge) to provide a clear fit between the data and what they represent (real world objects). Ontology provides the basis for understanding and documenting the real-world semantics of the data and a pathway for the data to be 'sophisticated'. Sophistication is a term that refers to the process of improvement along several dimensions which provides the pathway to evaluate the semantic richness of the data. The following section argues the case for a different approach by critically examining the current LIS renovation approaches. Section 3 explains the concept of CS. Section 4 describes the research design underlying the current work. Section 5 describes CS through an example in the context of the migration of a large financial system. Section 6 discusses the

theoretical and practical implications of the approach. Lastly, conclusions and directions for future research are presented.

## II. BACKGROUND AND RELATED RESEARCH

In the field of Information Systems (IS), the meaning attributed to LIS is normally confined to two different perspectives. One perspective interprets and defines LIS in a depreciative sense, suggesting an aging system, incapable of adapting and changing as required by modern-day emergent organizations [13] [14]. Another school of thought adopts a more positive interpretation and views LIS as a valuable source (in some cases the only source) of business knowledge, which serves as a precious resource for future improvements to the IS and the organization as a whole [15] [16]. This research takes a more positive interpretation of LIS and views it as an important source of business knowledge, which should be preserved for the well-being of the organization.

The increasing cost of managing LIS together with the need to preserve business knowledge has meant that renovating LIS has become an important research topic over the years. While a number of approaches have emerged to solve the LIS problem, practical solutions have been slow to emerge for a variety of reasons. Most of the approaches are guilty of only considering technology-based solutions for LIS renovation. They 'unlock' a system view of the business from one technology set, make some changes to aspects of data and/or behavior, and 'lock' the revised view into another technology set. This focus has limitations, as a technology dependent solution means that the whole process is geared towards building a LIS for the future [4]. Moreover it means that the business knowledge is always locked into a specific technology at any given time. So every time a renovation project is initiated, the organization has to either find mechanisms to move business knowledge onto the target system or risk the prospect of developing this business knowledge again from scratch. Taking a technology dependent view means that the knowledge becomes further entrenched within the technology and any program understanding task becomes that much more difficult to achieve.

Considering that LIS are poorly designed and documented, in many instances such knowledge is the only source of business information and hence is vital and should be preserved at any cost [3]. One of the primary and most visible sources of business knowledge is 'business data'. Given the wealth of information that the data holds about the various aspects of the business, it is often viewed as a fundamental source of business knowledge [17]. Although few data-based solutions have emerged in response to the increasing relevance of data, they remain largely ineffective in either preserving and improving the underlying business knowledge or providing a technology independent solution. The approaches in this camp only consider the basic process of migrating the data from one system to another: Where effort is undertaken to understanding the semantics, the focus is necessarily limited to making improvements from an application perspective (schemas and data normalization aspects) and not from a business perspective (real world reference). Consequently, issues such as integration and harmonization of application data from various LIS remain largely unresolved primarily due to lack of clear semantics and standardized data [5].

Current solutions rarely make use of models and architectures which allow for specifications to be laid out clearly and understood from a business perspective and used for future reference. As a result, there is hardly any attempt to extract the business knowledge and document it in business models so as to (a) explicitly capture and separate the business knowledge from the underlying technology and (b) enrich the knowledge. The following is a brief overview of current legacy renovation approaches. This overview is aimed at highlighting the technology-dependent view of current approaches and how they seriously fail to address the problem of delivering semantically rich models of legacy systems' business data. The approaches taken from the literature are as follows:

- **Big Bang or Cold Turkey Approach** [18]: This approach advocates re-developing the existing system from scratch using modern architectures and the latest technology. The LIS remains operational until the new system is developed and subsequently switched off. The approach while forward looking, involves a very substantial risk for the organization. The approach suffers from being primarily a technology-based solution and completely ignores the knowledge stored within the existing system. Moreover, the assumption that the new system will always be better in terms of efficiency, cost and functionality remains an open issue.
- **Wrapping Approach** [7]: Wrapping offers an easy and cost effective alternative to replacement. The approach works by hiding the complexity of the system through modern looking interfaces [19]. LIS can be wrapped at the functional level, data level or at the user interface level. While wrapping offers a cost effective alternative, it very much remains a short-term solution for LIS renovation. The approach solves the immediate problem of LIS, but in the long term wrapping means an additional layer of code to manage. Moreover, being completely dependent on technology implies that the layer will continue to grow whenever the current technology is replaced.
- **Chicken Little Approach** [2]: This approach emerges as a result of the increasing significance of the data aspect of the LIS and the need to preserve the data assets. The approach proposes an 11-step gateway-based migration approach for renovating both the application and the legacy database concurrently. As the approach involves migrating both the data and application concurrently, completing the whole process can become time-consuming. With technology changing at a rapid pace, this can become really tricky, if the target technology becomes obsolete over time. Moreover as the approach merely migrates the data without making any kind of sophistication (semantics or schematic), the actual benefits of the approach become limited overall.
- **Butterfly Approach** [20]: This is a data migration approach which prescribes that systems do not need to interoperate while the data is being migrated. The approach eliminates the need for gateways as the LIS is

rendered read-only, while the data is migrated to the target system. Manipulation of the data is managed through software called DAA (Data Access Allocator) and all the manipulations during the migration process are stored in auxiliary data stores called TempStores (TS). Chrysaliser, a piece of software then manages the migration the data from the LIS and TS to the target system. The Butterfly Approach requires that the skeleton of the target database be designed completely before the migration process can initiate. With systems that have little documentation, understanding the data schemas and then replicating them can involve considerable effort and time. This cost can then increase exponentially every time the system needs to be renovated. Moreover , even though the approach stresses on understanding the semantics of the data, the actual scope is limited to understanding the data schemas so that they can be redesigned in the target system.

- **Iterative Re-engineering Approach** [8]: The Iterative Re-engineering approach builds on the Chicken Little approach by applying a data re-engineering strategy to the whole approach. The approach is incremental and works by dividing the LIS in small components and identifying the components that need to be re-engineered. The data aspects related to components that need to be renovated are initially understood and their data structures re-engineered and migrated to the target technology along with real data. The component is then re-engineered using modern tools and techniques. While the approach is data-centric, the focus remains in developing the target technology. The purpose of understanding the data is only aimed at separating relevant data from data that is redundant and no longer needs to be migrated. Similar to the Butterfly approach, semantics are rarely understood from a real world perspective and knowledge rarely captured to provide a better understanding of descriptions and relationships. Moreover, being technology focused means that the whole process will have to be done again once the current system becomes a legacy.

In summary, as highlighted in Table 1 the renovation approaches available for LIS either tend to be very technology motivated or are limited to basic data migration with the primary motive of migrating the data from one system to another. In either case they remain largely incapable of capturing the business knowledge expressed through business data in a manner that allows such knowledge to evolve and benefit the organization. Although, the Butterfly and Iterative Approaches are more mature, as they highlight the need to understand the semantics of the data, the focus remains clearly embedded in developing the target technology and not the underlying business knowledge. To overcome the enduring problems associated with a technology focused view, the focus of LIS renovation needs to shift to a more model-based approach because: (a) the use of technology in itself is not sufficient to represent all the complexities underlying the business which can only be done from a real world perspective and (b) to explicitly capture the business knowledge hidden within business data in a technology independent fashion, thus allowing scope for improving and enriching the data. The paper proposes such an approach, called Content Sophistication, whose objective is to capture and deliver sophisticated business knowledge underlying the business data in a technology agnostic fashion.

## III. CONTENT SOPHISTICATION

The 'Content Sophistication' (CS) approach is aimed at extracting and documenting the business knowledge hidden within business data (in the form of type and individual level data) from LIS and improving this knowledge along several dimensions of 'sophistication'. Business knowledge refers to the real world objects that the business data describes. Sophistication is an improvement process undertaken to explicitly understand and document this business knowledge, thus providing scope for increase in the semantic richness of the business data by reducing complexity and increasing the potential functionality and interoperability. Sophistication is achieved via the careful use of ontology which provides a framework and a process to improve the semantic richness of the data by clearly identifying the objects that exist within the

TABLE I. CURRENT LIS RENOVATION APPROACHES

| **IS Renovation Approaches** → <br> **Features Available/Supported** ↓ | **Re-engineering Approaches** | | | **Wrapping** | **Re-development** |
|---|---|---|---|---|---|
| | ***Chicken Little Approach*** | ***Butterfly Methodology*** | ***Iterative Re-engineering*** | | |
| Technology Focused | √ | √ | √ | √ | √ |
| Data Relevance | √ Focuses only on the migration of data from one technology to another | √ Schema level | √ Data level | × | × |
| Semantic Focus | × | Linguistic and application Perspective | Linguistic and application perspective | × | × |
| Architectural and Modeling Focus | × | × | × | × | × |
| Business Knowledge | Remains embedded within technology as knowledge is transferred from one system to another | Remains embedded within technology as focus is on improving database design | Remains embedded within technology as focus is on improving database design | × | × |
| Process Structure | Incremental and iterative although the whole system needs to be transformed | Incremental and iterative | Incremental and iterative although the whole system needs to be transformed | Ad-hoc | Ad-hoc and complete transformation at one go |
| × - Does not consider or Feature not available <br> √ - Feature available in some form or other | | | | | |

business along with the types and relationships among those objects. This section provides an overview of the CS approach and articulates how the approach deals with the problem prevalent within current LIS renovation approaches.

The major limitation of current approaches lies in their inability to capture business knowledge at a technology independent level. Efforts are rarely aimed at capturing the business knowledge at a Computational Independent Model (CIM) level [21]. CIM is a part of the Object Management's Group's (OMG) Model Driven Architecture (MDA) initiative. The CIM level highlights the need for well-developed business models and aims to separate business concerns from application concerns and the underlying platform technology. Such a view is important as modern organizations are in a constant state of evolution and business requirements are continuously changing [22]. In order to understand and represent business requirements, it is important to capture the relative knowledge in models that allow an organization to understand the inherent composition of the business, along with a clearly defined way to deliver business value, irrespective of particular application concerns. CS operates at a CIM level, producing business models that are independent of any type of technology or platform. Not being tied to any specific technology, CS allows the organization to understand and document knowledge in terms of its business semantics providing scope for future refinements and re-use.

In addition to this model-based perspective (not present within the current LIS approaches), there is also the issue of clarifying the real world semantics of the data. In general, semantics is broadly defined around the notion of 'sense' and 'reference', which form the basis of a concept's (data) meaning [23]. While reference is the relationship between the concept and the objects it refers to in real world (real world semantics), sense is the thought that the concept expresses (reflected in a concept's relationships to other concepts). The issue with present approaches is that they intuitively recognize the need to understand the sense of the data (linguistic meaning) without clearly understanding what the data means from a real world perspective. This can have severe limitations as the sense perspective is often dependent on a number of assumptions and the context in which it is used or defined. These assumptions can depend on designer preferences, application needs or language used to define the concept and they rarely take into account the real world reference of the concept as this knowledge is often viewed as implied within the concept itself. Consequently, many systems often face heterogeneity issues, because the same concepts, which look semantically equivalent from a sense perspective, are different because they reference (or correspond to) different real world objects and vice-versa [24]. To clarify and remove semantic heterogeneity surrounding the data and its structure, it is often important to understand the business knowledge hidden behind the data, i.e., what objects they describe – their real world semantics. The use of ontology helps to clarify the real world semantics hidden within the data and also provides a framework within which the objects can be understood, modeled, and sophisticated to be used as a reference ontology for future applications.

CS is an ontology-based approach; it uses the philosophical notions of ontology as the basis for understanding, modeling and sophisticating the business data. The aim of philosophical ontology is to seek truth and develop theories that provide a clear description of what 'objects exist' in the real world of any domain, what relationships exist between the objects and their categorization (i.e., types) [25]. To decipher the business knowledge implicit in the data, it is important to understand the meaning of the data – its logical semantics. Ontology is used as an approach for understanding the semantics of the data, as it reduces the conceptual and terminological confusion, both implicit and explicit, and achieves a shared understanding [26] [27].

CS uses ontology at two levels. At an initial level, ontology provides a means for analyzing and deciphering the real world semantics that are hidden in the data. This is done through the notion of '*ontic commitment*'. Ontic commitment is based on 'what exists' and it expresses a commitment to the existence of certain objects and their categories. This can be explained through the following example. By nature, information is about something. More precisely, any IS (whether a business or computer system) refers to objects – and so implies that they exist. These objects are the information's ontic commitment. Applying the notion of ontic commitment is important as it allows moving away from the linguistic representation which is normally accepted as true to more valid claims of what exists. The basis for applying ontic commitment and developing the subsequent ontology comes from the data within the existing system. The existing data represents the underlying set of objects that exist in this world together with its relationships and types, thus providing the means for describing the ontology. At another level the relevant objects and relations of the ontology are modeled over time and across multiple systems and domains within a common conceptual framework (ontological model), which provides a representation of the ontology. The conceptual framework directly reflects the ontology, in other words objects modeled in the framework map to objects in the real world. While this representation not only removes any problems pertaining to implicit and hidden knowledge, it also allows the ontology developed to be shared with others who have similar needs for knowledge representation in that domain, thus saving a significant amount of labor that will have gone in replicating the same knowledge.

CS is based on the REV-ENG$^{TM}$ methodology [28], which defines a way to re-engineer existing systems into business models by using business objects and General Business Patterns (GBP). Business objects, as mentioned earlier, refer to the objects that exist in the real world and as a result have meaning and usage to the business (e.g., countries, postal regions). GBP are patterns of business objects that are related to each other (e.g., the pattern Geo-Political Region (GPR) expresses a relation between countries and postal regions amongst others). Both business objects and GBP are a 'sophisticated' representation of legacy business data that emerges through the process of applying CS to the LIS.

CS is normally carried out in two distinct phases, namely Interpretation and Sophistication. Interpretation is defined as identifying the business objects (e.g., countries, postal regions) that the system commits to existing (ontic commitment). Interpretation offers the opportunity to understand the real world semantics of the data from the application's perspective

and within its context of use. This knowledge might or might not concur with what exists in the real world, thus providing scope for sophisticating the business data and clarifying its real world semantics. The interpretation process works its way through the LIS identifying both the explicit and implicit business data. This underwrites the completeness relative to the LIS, ensuring that all the business data is captured in the final model. In outline terms, sophistication can be defined as the process of gradually improving a business model - by removing any discrepancy between the semantics of the data from an application perspective and what exists in the real (business) world. The aim is to provide better theories and a more precise representation of the world. Sophistication thus provides the underpinnings for stability and the evaluation of the stability aspect can be judged along the following dimensions:

- **Explanatory power**: The ability of the improved model can give increased meaning to the objects and the relationships expressed.
- **Fruitfulness**: The degree to which the improved model can meet currently unspecified requirements or is easily extendable to do so.
- **Generality**: The degree by which the scope of the types in the improved model can be increased without the loss of information.
- **Objectivity**: The ability of the model to provide a more objective (shared) understanding of the world.
- **Precision**: The ability of the improved model to give a more precise picture of the business object: in particular, to index a thing to its mode of existence as opposed to its mode of representation and/or application.
- **Simplicity**: The degree by which the model can be made less complex.

## IV. RESEARCH DESIGN

The research approach adopted in this work is based on grounded theory [29] [30]. A grounded theory approach to conducting the research was deemed appropriate. Firstly, given that ontology is about the objects that exist in the real world, a natural consequence is that work aimed at modeling ontologies should be based on empirical observation. Secondly, in the presence of large complex LIS (of the magnitude of tens of thousands of individual data items) a methodical approach to collecting, analyzing and identifying patterns in the data is necessary. Thirdly, the research described in this paper is novel. In the absence of previous theories on the subject area, a theory could only be constructed by grounding it in the data of the systems available.

The primary aim for using the grounded theory approach is the discovery of the conceptual framework. The process underlying the discovery is CS. The former can be viewed as the theory that emerges, whereas the latter represents the 'grounded' process for the discovery of the framework. As highlighted in Figure 1, the process undertaken to extract the business objects is grounded in the data of operational LIS. All GBP and objects in CS are grounded in data; none result from pure deduction. While the focus at present is to identify the necessary sophistication gaps and develop the initial framework, it is however necessary to test the generality of the framework across multiple systems and domains and refine the framework until a saturation point is reached (i.e., new data does not impact the model). It is only then that an agreement can be reached as to whether the ontological model that has been developed is complete and comprehensive. The nature of how general the framework can be needs to be considered from a pragmatic perspective.

While explaining the conceptual framework is outside the scope of this paper, the ontological constructs (metadata repository) and how CS has been applied to extract and sophisticate the business knowledge is defined in the next section. The sophisticated models that emerge as a result of applying the CS process only form a part of the overall conceptual framework alongside other sophisticated models [31].

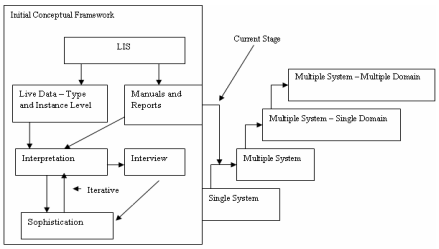


Figure 1. Research Design

## V. CONTENT SOPHISTICATION APPLIED

To highlight its practical benefits, CS is best explained through an example of a Sophistication Instance (SI). A SI is a CS artifact whose purpose is to enable the benefits of CS to be easily presented to and understood by a business audience. The SI presented in this section is part of a piece of work undertaken to renovate a large financial system, henceforth referred to as 'App X' and the company as 'Company X'. Company X is a large information technology solution and services company with interest in a number of vertical markets. App X is a large financial system developed, maintained and managed by Company X. The system originally written in state of the art technology around 20 years back is based on obsolete technology and difficult to manage today. Client feedback, market directions and the long-term business objectives of the company have resulted in a business goal to migrate the system towards modern technical platforms. This has presented a situation to model the systems in a technology agnostic fashion, thus insulating the risk of being dependent on any specific technology. The SI described here concentrates on the Geo-Political Regions (GPR) GBP, as it is regarded as simple enough to be easily understandable, while also being rich enough to demonstrate the feasibility of the approach. The GPR GBP has emerged by applying the CS process to the existing system. Before presenting the SI, the object paradigm will be presented in order to provide the basis for understanding the diagrams that follow.

The object paradigm [28], not to be confused with the object-oriented paradigm, provides the basis for the

representation of ontologies and therefore models real world objects. It provides a set of ontological constructs necessary for describing the structure of the real world. In the object paradigm, while everything is an object, objects themselves can be classified as (a) elements, (b) tuples, (c) types, or (d) tuple types. An element is an object that does not have instances (i.e., 'United Kingdom' and 'GB'). A tuple is a relationship between elements. For example, the coded by relationship between 'United Kingdom' and 'GB' is a tuple. A type is an object that has instances. For example, 'Country' and 'Country Codes' are types. An instance of 'Country' is 'United Kingdom' and an instance of 'Country Codes' is 'GB'. A tuple type is a type of relationship. For example, Countries are coded by Country Codes, hence 'coded by' represents all the tuples (relationships) in the world between Countries and Country codes. A tuple is an instance of a tuple type. The notation used to represent elements, tuple, types and tuple types is shown in Figure 2. Note that tuple types are labeled 'tuples' (plural) while tuple instances are labeled 'tuple' (singular). Furthermore, the dashed lines with arrowheads represent 'instance of' relationships between a type and an element.

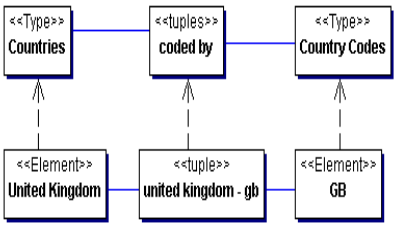


Figure 2. Object Paradigm Explained

The SI described here proceeds in three main steps: (1) Initial content, (2) Intermediate sophistication steps and (3) Final sophistication steps.

### A. *Intial Content*

The SI highlighted here concentrates on the sophistication of the CNTRY (Country Details) fragment. The interpretation process has revealed a sophistication gap between the Migrated Business Ontology and App X's Application Ontology. The CNTRY fragment as defined in the current system is used in the operational calculation of settlement days and in various reporting facilities (for e.g., limit reporting). There is however an implicit assumption by these operational and reporting facilities that the countries stored on the CNTRY fragment are disjoint. In reality however, there are countries that are not disjoint, the United Arab Emirates (UAE) containing Abu Dhabi is an example. This type of disjointness constraint is common in computer systems and is known as stratification. The disjointness constraint does not specify which countries are allowed to be set up in CNTRY, merely that whatever countries are set up must be disjoint. Where two countries overlap, the disjointness constraint allows either of the two to be set up, but not both. This can lead to different implementations having different incompatible CNTRY tables, even though the application level CNTRY description is identical. This situation is called implementation indexing – where the table is indexed to the implementation.

Figure 3 shows example interpretations of two possible implementations of App X, a Bank of England (BoE) country table and an ISO country table. BoE countries include Abu Dhabi and Dubai and ISO countries include the United Arab Emirates. The interpretation process has revealed a gap in the application's ontology. The process has revealed that even though both these tables refer to the same notion of countries, the two tables however cannot be combined as the tables do not cater for the scenario where the countries overlap (Abu Dhabi and Dubai are parts of the United Arab Emirates). This limitation is due to the stratified nature of CNTRY within the App X, which does not support nesting of countries.

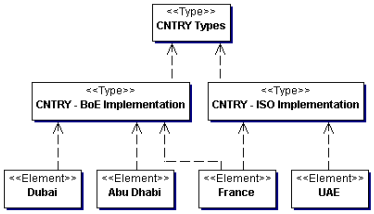


Figure 3. Legacy Representation of Stratified Countries

The above limitation can be highlighted using the following competency question:

- Can both nesting (e.g., United Arab Emirates) and nested countries (e.g. Abu Dhabi and Dubai) be represented?

A key question that the sophistication needs to address is how to provide a non-implementation indexed ontology that can combine these two implementation ontologies for countries.

### B. *Intermediate sophistication steps*

The common application level description of CNTRY fragment indicates that there is a common business pattern that links the application level with the different implementations. There is an implicit notion of countries, as the individual implementations of CNTRY are meant to be constrained to individual sets of countries. Figure 4 aims to highlight the stratified representation of countries in App X, by recognizing that App X's CNTRY commits to the existence of both CNTRY types (the collection of implemented CNTRYs) and Countries (as a super-type of every implemented CNTRY). Taken together these help to explain the common pattern of the two example implementations. To clarify things further and complete the pattern, *Stratified Country Types* and *Country Types* are introduced.

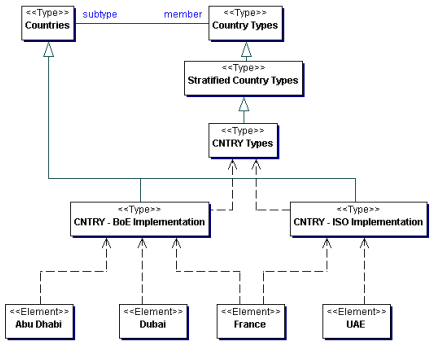


Figure 4. Country Stratified Types

A further sophistication step is shown in Figure 5. It eliminates the implementation indexed *CNTRY Types* as redundant – its work is done by *Stratified Country Types*. This enables multiple stratified country types in a single implementation (if required), as demonstrated by *BoE Stratified Countries* and *ISO Stratified Countries*.

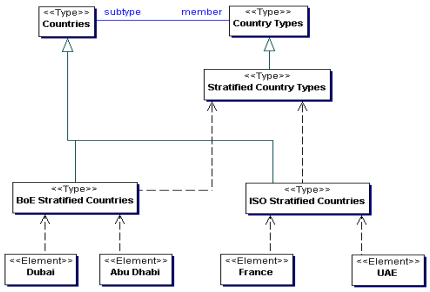


Figure 5. Country Stratified Types

## C. *Sophisticated model*

Figure 6 illustrates the (destratified) sophisticated ontology proposed for the business data. It illustrates how all the elements introduced in Figure 3 can fit into this ontology. To enable consolidated reporting, it is not sufficient to relax the constraint on disjointness. There needs to be a way to represent the overlapping. This is done by introducing a whole-part relationship – the example here is between United Arab Emirates and Abu Dhabi and Dubai- this allows the overlapping of countries to be shown and modeled explicitly.

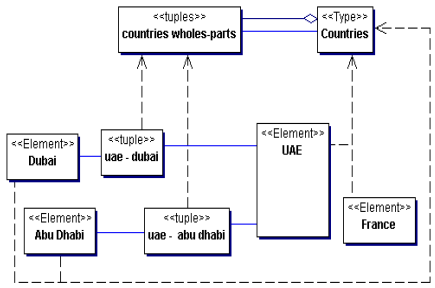


Figure 6. Destratified Nested Countries

For reference Table 2 indicates how the sophistication process has delivered improvements in terms of the dimensions highlighted in the previous section.

TABLE II. SOPHISTICATION DELIVERED ACROSS DIMENSIONS

| Sophistication Dimension | Delivered Sophistication |
|---|---|
| Objectivity | The resulting model is not implementation indexed as was the case with the original App X system. |
| Generality | At a later stage, the country wholes-parts relationship is generalised to GPR. This is not shown in this SI given that it is not within the scope of the fragment's data. |
| Simplicity | It is simpler in that it no longer needs to deal with the stratified country patterns. |
| Precision | The more general *Countries* are a more precise reflection of the common-sense notion of a country than the specialised implementation indexed App X Countries. |
| Explanatory power | It has increased the explanatory power, as it has explicitly modeled the whole-part relationship between UAE and Abu Dhabi. |

# VI. IMPLICATIONS FOR THEORY AND PRACTICE

The critical analysis of existing renovation approaches has highlighted important gaps. The near dependence on technology, negligence towards business data, coupled with issues like abstractness and limited practical ability of the existing approaches has clearly highlighted the need for a different approach for legacy renovation. In light of these arguments the theoretical and practical implications of CS can be summarized as follows:

- **An Ontological Approach to developing IS**: CS provides a different approach to developing and designing IS. Use of ontology clarifies object semantics and ensures that all the objects and their relationships are explicitly identified and reflect real world concerns.
- **Technology Agnostic Development**: To avoid creating 'legacy of the future', renovation approaches must focus at the CIM level. CS clearly underscores the need for well-developed business models and aims to separate business and/or application concerns from underlying platform technology, with the objective of making business models explicit and allowing them to evolve in line with business needs (as opposed to technology).
- **Semantically rich data for future business needs**: The literature is awash with instances highlighting the importance of business data stored within systems [32]. While CS is not only data-driven, it progressively aims to make the data richer and more sophisticated in order to cater for future requirements. With reference to Section 5, country destratification allows to explicitly represent nested countries. In this specific example it was not possible to generalize Country to GPR as no non-country data was present to drive this. However, in other SI examples, where non-country data is available (e.g., Postal Regions, Country Groups), Countries have been generalized to GPR and Country Whole-Parts to GPR Whole-Parts [31]. This generalization has two distinct advantages. While there is no loss in the semantic richness of the data, generalizing reduces the number of objects to manage.
- **Data richness induces better reuse and interoperability**: CS recognizes that using ontology as a basis for sophistication can improve the real world semantics of the data. The sophistication of the CNTRY fragment has resulted in a more general notion of Countries, which is a more precise reflection of the common sense notion of a

country than the specialized implementation indexed App X Countries. Removing the implementation-indexed constraint has allowed country representations of different implementations to be interoperable. For example, UAE, Abu Dhabi and Dubai can now be loaded onto a single Country Table. Moreover, since the business models reflect a real world scenario, these models can be readily (re)used in any organization or across domains. For example, loading the sophisticated notion of countries can only enhance the functionality provided by the system.

- **Flexible, Incremental and Effective**: CS is completely flexible, as it is not tied to any particular migration tool. Any design tool, depending on the needs and familiarity can be used for CS. Moreover, segmenting the system allows the migration to be done incrementally, which then can be tested thoroughly using competency questions and validation queries on the sophisticated data. The effectiveness of the approach is proportional to the scope and the amount of time spent in analyzing the existing system. Increases in scope can lead to better opportunities for sophistication of the business data, which over time provide a more significant payback as economies of scope and scale are increasingly achieved.

The success of CS is affected by several factors including the availability of both type and individual level data, business and technical expertise of the existing system, and business modeling knowledge. The adoption of the CS approach is coupled with a necessary learning curve. This learning curve is however necessary as the approach requires a new way of thinking about systems development and so new development rules apply and new competences are required. Moreover, the level of sophistication and its associated benefits are directly proportional to the application of the CS approach within real systems. Since the approach is firmly grounded in data, the benefits are greater when the models are generalized out of several systems.

## VII. CONCLUSION

CS provides a business perspective to LIS renovation. Such a perspective has been lacking within present renovation approaches which merely focus on renovating the technical aspect of the LIS. A primarily technical focus can become an enduring problem as the whole effort is spent on building a LIS for the future. This is compounded by the fact that majority of the approaches ignore the business knowledge hidden within these systems, which is often the most stable and important asset of the organization. No efforts are spent on extracting and documenting this knowledge in a way that allows the organization to have a clear representation of objects that are important to the business.

The aim of CS is to overcome the above problems by capturing the business knowledge hidden within the system in a technology agnostic fashion. Being independent of any application or platform concerns ensures that the outcomes of CS can outlive the life of one renovation project and can subsequently be used for future renovation projects. While the premise of the approach is to capture the business knowledge, an earnest effort is also made to make sure that the knowledge that is captured is consistent with the objects that exist in the real world. Ontology provides the basis for such a reflection as it provides a clear description of the objects that exist in this world through the notion of ontic commitment. The resulting benefits of an ontology focused approach are improved semantics, better interoperability and less complexity.

The benefits of CS have been demonstrated here through an example of a sophistication instance (an artifact delivered by the CS process). Among others, benefits have been achieved in a reduction in complexity and an increase in potential functionality and interoperability. Future research will focus on applying the GPR framework across multiple systems to test the generality of the framework and make the necessary refinements. Work will also continue in developing frameworks where new GBP have been identified (e.g. a Product GBP). This iterative approach will allow the CS process to mature and at the same time test the resilience of patterns already identified. Further work will also focus on the development of a software tool to automate CS activities as well as combining CS with application-level development.

## REFERENCES

[1] Bennett, K., "Legacy systems: coping with success", IEEE Software, vol 12(1), January 1995, pp. 19-23.

[2] Brodie, M. and M. Stonebraker, Migrating Legacy Systems: Gateways, Interfaces and the Incremental Approach. San Francisco: Morgan Kaufmann Publishers Inc., 1995.

[3]Young-Gul, K., "Improving Legacy systems maintainability", Information Systems Management, pp. 7-11, 1997.

[4] Kelly, S., N. Gibson, C. P. Holland and B. Light, "A business perspective of legacy information systems" Communications of the AIS, vol 2(1), pp. 1-27, July 1999.

[5] Aiken, P.H., Y. Yoon, and B. Leong-Hong, "Requirements-driven data engineering", Information & Management, vol 35(3), pp. 155-168, March 1999.

[6] Bisbal, J., D. Lawless, B. Wu and J. Grimson, "Legacy Information Systems: Issues and directions" IEEE Software, vol 16(5), pp. 103-111, September/October 1999.

[7] Comella-Dorda, S., K. Wallnau, R. C. Seacord and J. Roberts, "A Survey of Legacy modernisation approaches", Carnegie Mellon University, Software Engineering Institute, pp. 1-20, April 2000.

[8] Bianchi, A., D. Caivano, V. Marengo and G. Vissagio, "Iterative reengineering of Legacy Systems", IEEE Transactions on Software Engineering, vol 29(3), pp. 225-241, March 2003.

[9] Ganti, N. and W. Brayman, Transition of Legacy Systems to a Distributed Architecture. New York: John Wiley and Sons, 1995.

[10] Tilley, S.R. and D.B. Smith, Perspectives on Legacy Systems reengineering. Carnegie Mellon University. Pittsburg: Software Engineering Institute, 1995.

[11] Fayed, M. and S. Wu, "Merging multiple conventional models in one stable mode", Communications of ACM, vol 45(9), pp. 102-106, September 2002.

[12] Eriksson, H. and M. Penker, Business modeling with UML: Business Patterns at work, New York: John Wiley and Sons, 2000.

[13] Ulrich, W., "From Legacy Systems to Strategic Architectures", Software Engineering Strategies, vol 2(1), pp. 18-30, March/ April 1994.

[14] Connall, D. and D. Burns, "Reverse Engineering: Getting a grip on Legacy Systems", Data Management Review, vol 2(1), pp. 24-27, 1993.

[15] O'Callaghan, A.J., "Migrating large-scale Legacy Systems to Component-based and Object technology: the evolution of a pattern language", Communications of the AIS, vol 2(1), pp. 1-43, July 1999.

[16] Weill, P. and M. Broadbent, Leveraging, The new infrastructure: How market leaders capitalize on Information Technology. Boston: Harvard Business School Press, 1998.

[17] Robertson, P., "Integrating Legacy Systems with modern corporate applications", Communications of ACM, vol 40(5), pp. 39-46, May 1997.

[18] Bateman, A. and J. Murphy, Migration of Legacy Systems, School of Computer Applications. Dublin: Dublin City University: Dublin, 1994.

[19] Winsberg, P., "Legacy Code: Don't Bag it, Wrap it",. Datamation, vol 41(9), pp. 36-41, 1995.

[20] Wu, B., D. Lawless, J. Bisbal, R. Richardson, J. Grimson, V. Wade and D. O'Sullivan, "The Butterfly Methodology: A gateway-free approach for migrating Legacy Information Systems", In Proceedings of the ICECCS97, Villa Olmo:Italy, 1997.

[21] Miller, J. and J. Mukerji, MDA Guide Version 1.0., OMG, 2003 (http://www.omg.org/docs/omg/03-05-01.pdf (June 11, 2004).

[22] Truex, D.P., R. Baskerville, and H. Klien, "Growing systems in emergent Organizations", Communications of ACM, vol 42(8), pp. 117-123, August 1999.

[23] Frege, G., The foundation of Arithmetic: A logico-mathematical enquiry into the concept of number. 1884.

[24] Sheth, A. and J. Larson, "Federated Database Systems for managing distributed, heterogeneous and autonomous Databases", ACM Computing Surveys, vol 22(3), pp. 183 - 236, September 1990.

[25] Smith, B., Ontology and Information Systems, 2000 (http://wings.buffalo.edu/philosophy/faculty/smith/articles/ontologies.htm (June 11, 2004).

[26] Uschold, M. and M. Gruniger, "Ontologies: Principles, methods and applications", Knowledge Engineering Review, vol 11(2), pp. 93-155, 1996.

[27] Chandrasekaran, B., J. Josepheson, and V.R. Benjamins, "Ontologies: What are they? Why do we need them?", IEEE Intelligent Systems and Their Applications, vol 14(1), pp. 20-16, 1999.

[28] Partridge, C., Business Objects: Re-Engineering for Reuse. Oxford: Butterworth-Heinemann, 1996.

[29] Glaser, B.G. and A.L. Strauss, The Discovery of Grounded Theory: Strategies for qualitative research, London:Weidenfeld and Nicholson, 1967.

[30] Turner, B.A., "The use of grounded theory for qualitative analysis of organizational behaviour", Journal of Management Studies, vol 20(3), pp. 333-348, 1983.

[31] Daga A., S. de Cesare, M. Lycett and C. Partridge, "Software Stability: Recovering general patterns of business", In Proceedings of the AMCIS'04, New York, USA, 06-08 August, 2004.

[32] Coyle, F.P., "Legacy integration changing perspectives", IEEE Software, vol 17(2), pp. 37-41, March/April 2000.